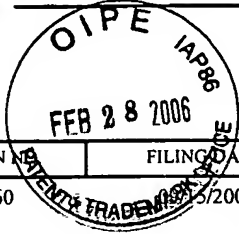


Ifw



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov



APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/662,560	02/15/2006	Naoyuki Hatano	9281-4635	9744
35928 7590 02/15/2006				
DLA PIPER RUDNICK GRAY CARY US, LLP 1625 MASSACHUSETTS AVENUE, NW SUITE 300 WASHINGTON, DC 20036-2247				
EXAMINER CHEN, ALAN S				
ART UNIT PAPER NUMBER				
2182				

DLA PIPER RUDNICK
GRAY CARY US LLP

FEB 24 2006

DATE MAILED: 02/15/2006

Please find below and/or attached an Office communication concerning this application or proceeding.



Office Action Summary

Application No. 10/662,560	Applicant(s) HATANO, NAOYUKI	
Examiner Alan S. Chen	Art Unit 2182	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 15 September 2003.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-4 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-4 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 15 September 2003 is/are: a) ☐ accepted or b) ☒ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. _____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|---|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input checked="" type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date <u>09/15/2003</u> . | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

Drawings

1. The drawings are objected to because Fig. 1 lacks textual labels that would greatly assist and understanding the invention. Corrected drawing sheets in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The figure or figure number of an amended drawing should not be labeled as "amended." If a drawing figure is to be canceled, the appropriate figure must be removed from the replacement sheet, and where necessary, the remaining figures must be renumbered and appropriate changes made to the brief description of the several views of the drawings for consistency. Additional replacement sheets may be necessary to show the renumbering of the remaining figures. Each drawing sheet submitted after the filing date of an application must be labeled in the top margin as either "Replacement Sheet" or "New Sheet" pursuant to 37 CFR 1.121(d). If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

3. Claims 1-3 are rejected under 35 U.S.C. 102(e) as being anticipated by US Pat. No. 6,678,728 to Uppunda et al. (Uppunda).

4. Per claims 1 and 2, Uppunda discloses a communication control device (Fig. 1 and 2, element 106 is the NIC) for controlling data communication between a host computer and a peripheral device (Fig. 1, ASIC controls communication between device on bus 120, here construed as the “host” devices and the NIC, element 106, is the peripheral device), comprising: first storage means, FIFO (Fig. 2, element 208), for storing data to be sent to the host computer (data stored in FIFO is awaiting transfer output of the NIC 106 to hosts on bus 120); second storage means for storing data outputted from the peripheral device (Fig. 2, element 206 gets data from devices on bus 118, e.g., the processor; note that it is expressly stated in Column 3, line 1+ that); and a control unit (Fig. 2, element 202) for transferring the data stored in the second storage means to the first storage means (Column 3, lines 20+, ASIC controls operation of NIC, where once the PC is awakened, data from the second/receive FIFO 206 will be sent to the first/transmit FIFO 208, either directly or from the EEPROM 204, all being controlled by the ASIC), when receiving a transmission approval command of approving data transmission from the peripheral device to the host computer (the transmission approval command is directly reflected in as the wake-up signal that the ASIC receives, Column 3, lines 54, “...a signal is generated is generated to cause the PC 122 to exit the sleep state so that the required action may be taken.” Note that Uppunda anticipates the type of scenario that the instant application is attempting to solve where a device is in a sleep state and has the chance of failing to receive certain data sent to it while in a sleep state.

Art Unit: 2182

5. Per claim 3, Uppunda discloses claim 1 wherein the second storage means is a multi-stage FIFO buffer (FIFO shown in Fig. 2 has multiple data elements, hence multi-staged).

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

7. The factual inquiries set forth in *Graham v. John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

8. Claim 4 is rejected under 35 USC 103(a) as being unpatentable over Uppunda in view of Wikipedia->USB.

Uppunda discloses Claim 1. Uppunda further discloses the NIC, element 106, being an external device, e.g., one that sits between the PC (Column 3, lines 1+, "NIC 106 could be external to the PC 122...", in the context of this rejection, the PC would be a peripheral since data is being sent from it to the devices on the network, i.e., the hosts) and it may use a different bus architecture, clearly one for connecting external devices.

Uppunda does not disclose expressly using the USB bus standard for the bus architecture for the NIC connecting to the PC.

Wikipedia discloses that USB is the one of the *most* prevalent bus standards to connect to external devices.

At the time of the invention it would have been obvious to a person of ordinary skill in the art to use USB for making connection between PC and NIC.

The suggestion/motivation for doing so would have been USB provides high bandwidth and a very well known industrial bus standard.

Therefore, it would have been obvious to combine Uppunda with Wikipedia->USB for the benefit of utilizing a well-known and fast external bus standard.

Conclusion

9. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Patents and patent related publications are cited in the Notice of References Cited (Form PTO-892) attached to this action to further show the state of the art with respect to transmit and receive buffers prevent loss of data during sleep modes.


10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Alan S. Chen whose telephone number is 571-272-4143. The examiner can normally be reached on M-F 8:30am - 5:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kim N. Huynh can be reached on (571) 272-4147. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

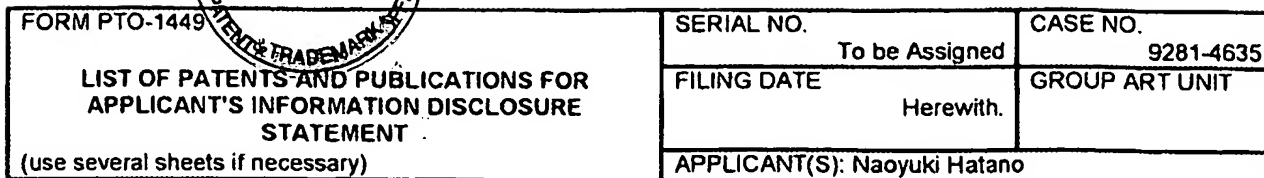
Art Unit: 2182

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

ASC
01/26/2006



KIM HUYNH
SUPERVISORY PATENT EXAMINER
2/2/06




U.S. PATENT DOCUMENTS

[illegible]

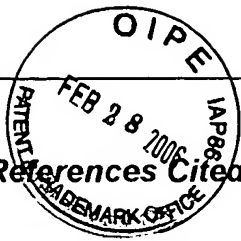
FOREIGN PATENT DOCUMENTS

EXAMINER INITIAL		DOCUMENT NUMBER	DATE	COUNTRY	CLASS/ SUBCLASS	TRANSLATION YES	NO
ASC	A3	10-145435	05/1998	Japan		X	

[illegible]

EXAMINER 	DATE CONSIDERED 1/26/86
--	-------------------------

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609; Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.



Notice of References Cited

Application/Control No.

10/662,560

Applicant(s)/Patent Under
Reexamination
HATANO, NAOYUKI

Examiner

Alan S. Chen

Art Unit

2182

Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-6,678,728 B1	01-2004	Uppunda et al.	709/222
*	B	US-6,990,676 B1	01-2006	Proehl et al.	725/40
*	C	US-6,134,665 A	10-2000	Klein et al.	713/300
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

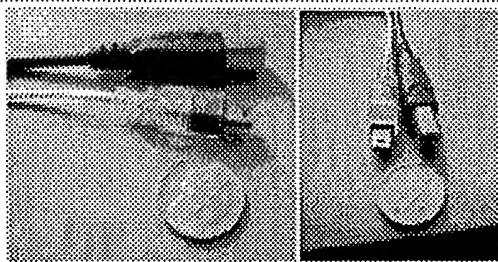
*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	www.wikipedia.org, search term="USB"□□
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

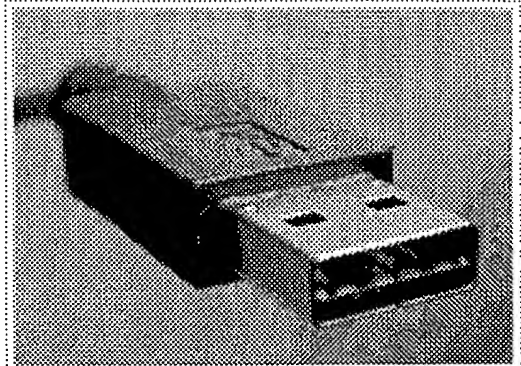
Universal Serial Bus

From Wikipedia, the free encyclopedia.
(Redirected from USB)

For other meanings of the abbreviation USB see USB (disambiguation).



Dual images of the two Type B USB connectors, mini and full size, side and front view, compared with a U.S. 5¢ piece (nickel) in both images for scale.



Type A USB connector

Universal Serial Bus (USB)

provides a serial bus standard for

connecting devices, usually to computers such as PCs and the Apple Macintosh, but is also becoming commonplace on video game consoles such as Sony's PlayStation 2, Microsoft's Xbox 360, Nintendo's Revolution, and PDAs, and even devices like televisions and home stereo equipment.

Contents

- 1 Overview
 - 1.1 Standardization
- 2 Technical details
 - 2.1 Device classes
 - 2.2 USB signaling
 - 2.2.1 Standard USB signaling
 - 2.2.1.1 Transfer speed
 - 2.2.2 Mini USB signaling
 - 2.3 USB connectors
 - 2.4 Power supply
- 3 USB compared to other standards
 - 3.1 Storage
 - 3.2 Human-interface devices (HIDs)
 - 3.3 FireWire
 - 3.3.1 Technical differences
 - 3.4 USB 2.0 Hi-Speed vs FireWire
- 4 Version history
 - 4.1 USB
 - 4.2 USB On-The-Go Supplement
 - 4.3 Wireless USB
- 5 Extensions to USB
- 6 Communication with USB devices
 - 6.1 Communication from the Linux OS
 - 6.2 Communication from the Mac OS
 - 6.3 Communication from the Windows OS

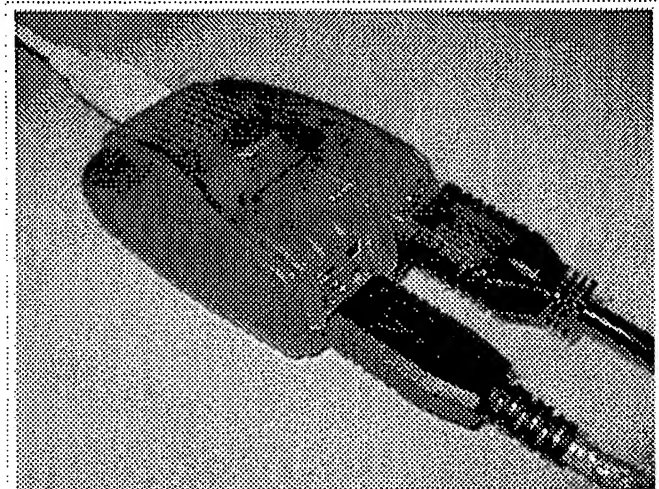
- 6.4 Other communication options
- 7 See also
- 8 External links

Overview

A USB system has an asymmetric design, consisting of a host controller and multiple devices connected in a tree-like fashion using special hub devices, called USB hubs. There is a limit of 5 levels of branching hubs per controller. Up to 127 devices may be connected to a single host controller, but the count must include the hub devices as well. A modern computer likely has several host controllers so the total useful number of connected devices is beyond what could reasonably be connected to a single controller. There is no need for a terminator on any USB bus, as there is for SPI-SCSI and some others.

The design of USB aimed to remove the need for adding separate expansion cards into the computer's ISA or PCI bus, and improve plug-and-play capabilities by allowing devices to be hot swapped (or added to the system without rebooting the computer). When the new device first plugs in, the host enumerates it and loads the device driver necessary to run it.

USB can connect peripherals such as mice, keyboards, gamepads and joysticks, scanners, digital cameras, printers, hard disks, and networking components. For multimedia devices such as scanners and digital cameras, USB has become the standard connection method. For printers, USB has also grown in popularity and started displacing parallel ports because USB makes it simple to add more than one printer to a computer. As of 2004 there were about 1 billion USB devices in the world. As of 2005, the only large classes of peripherals that cannot use USB (because they need a higher data rate than USB can provide) are displays and monitors, data acquisition devices that use FireWire ports, and high-quality digital video components.



USB hub

Standardization

The design of USB is standardized by the USB Implementers Forum (USB-IF), an industry standards body incorporating leading companies from the computer and electronics industries. Notable members have included Apple Computer, Hewlett-Packard, NEC, Microsoft, Intel, and Agere.

The USB specification is at version 2.0 as of January 2006. Hewlett-Packard, Intel, Lucent, Microsoft, NEC and Philips jointly led the initiative to develop a higher data transfer rate than the 1.1 specification to meet the bandwidth demands of developing technologies. The USB 2.0 specification was released in April 2000 and was standardized by the USB-IF at the end of 2001. Previous notable releases of the specification were 0.9, 1.0, and 1.1. Each iteration of the standard is completely backward compatible with previous versions.

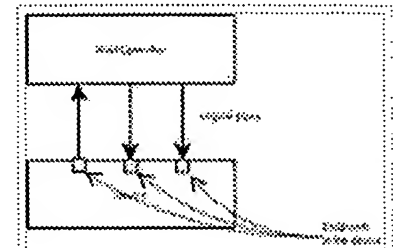
Smaller USB plugs and receptors called **Mini-A** and **Mini-B** are also available, as specified by the **On-The-Go Supplement to the USB 2.0 Specification**. The specification is of revision 1.0a currently.

Technical details

USB connects several devices to a host controller through a chain of hubs. In USB terminology devices are referred to as *functions*, because in theory what we know as a device may actually host several functions, such as a router that is a Secure Digital Card reader at the same time. The hubs are special purpose devices that are officially considered functions. There always exists one hub known as the root hub, which is attached directly to the host controller.

These devices/functions (and hubs) have associated *pipes* (logical channels) which are connections from the host controller to a logical entity on the device named an *endpoint*. The pipes are synonymous to byte streams such as in the pipelines of Unix, however in USB lingo the term *endpoint* is (sloppily) used as a synonym for the entire pipe, even in the standard documentation.

These endpoints (and their respective pipes) are numbered 0-15 in each direction, so a device/function can have up to 32 active pipes, 16 inward and 16 outward. (The *OUT* direction shall be interpreted *out of the host controller* and the *IN* direction is *into the host controller*.) Endpoint 0 is however reserved for the bus management in both directions and thus takes up two of the 32 endpoints. In these pipes, data is transferred in packets of varying length. Each pipe has a maximum packet length, typically 2^n bytes, so a USB packet will often contain something on the order of 8, 16, 32, 64, 128, 256, 512 or 1024 bytes.



USB endpoints actually reside on the connected device: the channels to the host is referred to as a pipe

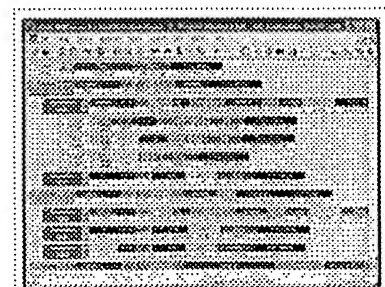
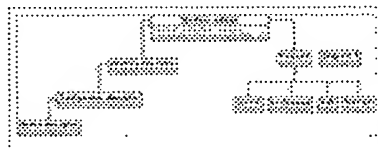
Each endpoint can transfer data in one direction only, either into or out of the device/function, so each pipe is unidirectional. All USB devices have at least two such pipes/endpoints: namely endpoint 0 which is used to control the device on the bus. There is always an inward and an outward pipe numbered 0 on each device. The pipes are also divided into four different categories by way of their *transfer type*:

- *control transfers* - typically used for short, simple commands to the device, and a status response, used e.g. by the bus control pipe number 0
- *isochronous transfers* - at some guaranteed speed (often but not necessarily as fast as possible) but with possible data loss, e.g. realtime audio or video
- *interrupt transfers* - devices that need guaranteed quick responses (bounded latency), e.g. pointing devices and keyboards
- *bulk transfers* - large sporadic transfers using all remaining available bandwidth (but with no guarantees on bandwidth or latency), e.g. file transfers

When a device (function) or hub is attached to the host controller through any hub on the bus, it is given a unique 7 bit address on the bus by the host controller.

The host controller then polls the bus for traffic, usually in a round-robin fashion, so no device can transfer any data on the bus without explicit request from the host controller. The *interrupt transfers* on corresponding endpoints does not actually interrupt any traffic on the bus, they are just scheduled to be queried more often and inbetween any other large transfers, thus "interrupt traffic" on a USB bus is really only high-priority traffic.

To access an endpoint, a hierarchical configuration must be obtained. The device connected to the bus has one (and only one) *device descriptor* which in turn has one or more *configuration*



USB Enumeration Trace

descriptors. These configurations often correspond to states, e.g. active vs. low power mode. Each configuration descriptor in turn has one or more *interface descriptors*, which describe certain aspects of the device, so that it may be used for different purposes: for example, a camera may have both audio and video interfaces. These interface descriptors in turn have one *default interface setting* and possibly more *alternate interface settings* which in turn have *endpoint descriptors*, as outlined above. An endpoint may however be reused among several interfaces and alternate interface settings.

USB device descriptors are hierarchical and quite complex. This UML diagram tries to give a entity relation between the different descriptors: the lower left device descriptor is highest in the hierarchy, this has configuration descriptors, which have interface descriptors, which have interface settings which in turn hold the actual endpoints.

The hardware that contains the host controller and the root hub has an interface toward the programmer which is called *Host Controller Device* (HCD) and is defined by the hardware implementer. In practice, these are hardware registers (ports) in the computer.

At version 1.0 and 1.1 there were two competing HCD implementations. Compaq's *Open Host Controller Interface* (OHCI) was adopted as the standard by the USB-IF. However, Intel subsequently created a specification they called the *Universal Host Controller Interface* (UHCI) and insisted other implementers pay to license and implement UHCI. VIA Technologies licensed the UHCI standard from Intel; all other chipset implementers use OHCI. The main difference between OHCI and UHCI is the fact that UHCI is more software-driven than OHCI is, making UHCI slightly more processor-intensive but cheaper to implement (excluding the license fees). The dueling implementations forced operating system vendors and hardware vendors to develop and test on both implementations which increased cost. During the design phase of USB 2.0 the USB-IF insisted on only one implementation. The USB 2.0 HCD implementation is called the *Extended Host Controller Interface* (EHCI). Only EHCI can support high-speed transfers. Each EHCI controller contains four virtual HCD implementations to support Full Speed and Low Speed devices. The virtual HCD on Intel and Via EHCI controllers are UHCI. All other vendors use virtual OHCI controllers.

On Microsoft Windows platforms, one can tell whether a USB port is version 2.0 by opening the Device Manager and checking for the word "Enhanced" in its description; only USB 2.0 drivers will contain the word "Enhanced." On Linux systems, the `lspci -v` command will list all PCI devices, and a controllers will be named OHCI, UHCI or EHCI respectively, which is also the case in the Mac OS X system profiler. On BSD systems, `dmesg` will show the detailed information hierarchy.

Device classes

Devices that attach to the bus can be full-custom devices requiring a full-custom device driver to be used, or may belong to a *device class*. These classes define an expected behaviour in terms of device and interface descriptors so that the same device driver may be used for any device that claims to be a member of a certain class. An operating system is supposed to implement all device classes so as to provide generic drivers for any USB device.

Device classes are decided upon by the Device Working Group of the USB Implementers Forum. If the class is to be set for the entire device, the number is assigned to the *bDeviceClass* field of the device descriptor, and if it is to be set for a single interface on a device, it is assigned to the *bInterfaceClass* field of the interface descriptor. Both of these are a single byte each, so a maximum of 253 different device classes are possible (values 0x00 and 0xff are reserved). If *bDeviceClass* is set to 0x00, the operating system will look at *bInterfaceClass* of each interface to determine the device class.

Each class also optionally supports a *SubClass* and *Protocol* subdefinition. These can be used as the main device classes are continuously revised.

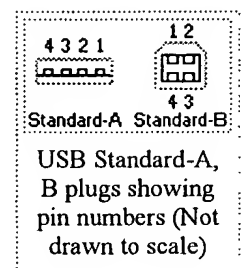
The most used device classes (grouped by assigned class ID) are:

0x00	Reserved value - used in the device descriptor to signify that the interface descriptor holds the device class identifier for each interface.
0x01	USB audio device class, sound card-like devices.
0x03	USB human interface device class ("HID"), keyboards, mice, etc.
0x06	Still image capture device class, identical to the Picture Transfer Protocol as used across USB
0x07	USB printer device class, printer-like devices.
0x08	USB mass storage device class used for flash drives, portable hard drives, memory card readers, digital cameras, digital audio players etc. This device class presents the device as a block device (almost always used to store a file system).
0x09	USB hubs.
0x0a	USB communications device class used for modems, network cards, ISDN connections, Fax.
0x0e	USB video device class, webcam-like devices, motion image capture devices.
0xe0	Wireless controllers, for example Bluetooth dongles.
0xff	Custom device class - used to establish that a device or interface does not support any standard device class and requires custom drivers.

USB signaling

Standard USB signaling

Standard USB connector pinout		
Pin	Function (host)	Function (device)
1	V _{BUS} (4.75–5.25 V)	V _{BUS} (4.4–5.25 V)
2	D [−]	D [−]
3	D ⁺	D ⁺
4	Ground	Ground



USB signals are transmitted on a twisted pair of data cables, labelled D⁺ and D[−]. These collectively use half-duplex differential signaling to combat the effects of electromagnetic noise on longer lines. D⁺ and D[−] operate together; they are not separate simplex connections.

Transfer speed

USB supports three data rates.

- A **Low Speed** rate of 1.5 Mbit/s (183 KiB/s) that is mostly used for Human Interface Devices (HID) such as keyboards, mice and joysticks.
- A **Full Speed** rate of 12 Mbit/s (1.4 MiB/s). Full Speed was the fastest rate before the USB 2.0 specification and many devices fall back to Full Speed. Full Speed devices divide the USB bandwidth between them in a first-come first-served basis and it is not uncommon to run out of bandwidth with several isochronous devices. All USB Hubs support Full Speed.
- A **Hi-Speed** rate of 480 Mbit/s (57 MiB/s), commonly called USB 2.0.

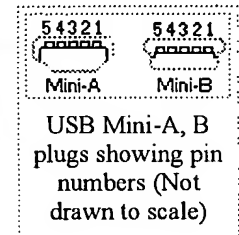
Not all USB 2.0 devices are Hi-Speed. A USB device should specify the speed it will use by correct labeling on the box it came in or sometimes on the device itself. The USB-IF certifies devices and provides licenses to use special marketing logos for either "Basic-Speed" (low and full) or High-Speed after passing a compliancy test and paying a licensing fee.

Hi-Speed devices should fall back to the slower data rate of Full Speed when plugged into a Full Speed hub. Hi-Speed hubs have a special function called the **Transaction Translator** that segregates Full Speed and Low Speed bus traffic from Hi-Speed traffic. The Transaction Translator in a Hi-Speed hub (or possibly each port depending on the electrical design) will function as a completely separate Full Speed bus to Full Speed and Low Speed devices attached to it. This segregation is for bandwidth only; bus rules about power and hub depth still apply.

Mini USB signaling

Mini USB connector
pinout

Pin	Function
1	V _{BUS} (4.4–5.25 V)
2	D ⁻
3	D ⁺
4	ID
5	Ground



Most of the pins of a mini USB connector are the same as a standard USB connector, except pin 4. Pin 4 is called ID and is connected to pin 5 for a mini-A to indicate which device should initially act as host, in the mini B this is open circuit. The Mini A also has an additional piece of plastic inside to prevent insertion into slave only device.

USB connectors

The connectors which the USB committee specified were designed to support a number of USB's underlying goals, and to reflect lessons learned from the varied menagerie of connectors then in service. In particular:

- The connectors are designed to be robust. Many previous connector designs were fragile, with pins or other delicate components prone to bending or breaking, even with the application of only very modest force. The electrical contacts in a USB connector are protected by an adjacent plastic tongue, and the entire connecting assembly is further protected by an enclosing metal sheath. As a result USB connectors can safely be handled, inserted, and removed, even by a small child. The encasing sheath and the tough moulded plug body mean that a connector can be dropped, stepped upon, even crushed or struck, all without damage; a considerable degree of force is needed to significantly damage a USB connector.
- It is difficult to incorrectly attach a USB connector. Connectors cannot be plugged-in upside down, and it is clear from the appearance and kinesthetic sensation of making a connection when the plug and socket are

correctly mated.

- The connectors are particularly cheap to manufacture.
- The connectors enforce the directed topology of a USB network. USB does not support cyclical networks, so the connectors from incompatible USB devices are themselves incompatible. Unlike other communications systems (e.g. RJ-45 cabling) gender-changers are never used, making it difficult to create a cyclic USB network.
- A moderate insertion/removal force is specified. USB cables and small USB devices are held in place by the gripping force from the receptacle (without the need for the screws, clips, or thumbturns other connectors require). The force needed to make or break a connection is modest, allowing connections to be made in awkward circumstances or by those with motor disabilities.
- The connector construction always ensures that the external sheath on the plug contacts with its counterpart in the receptacle before the four connectors within are connected. This sheath is typically connected to the system ground, allowing otherwise damaging static charges to be safely discharged by this route (rather than via delicate electronic components). This means of enclosure also means that there is a (moderate) degree of protection from electromagnetic interference afforded to the USB signal while it travels through the mated connector pair (this is the only location when the otherwise twisted data pair must travel a distance in parallel).
- The USB standard specifies relatively low tolerances for compliant USB connectors, intending to minimize incompatibilities in connectors produced by different vendors (a goal that has been very successfully achieved). Unlike most other connector standards, the USB spec also defines limits to the size of a connecting device in the area around its plug. This was done to avoid circumstances where a device complied with the connector specification but its large size blocked adjacent ports. Compliant devices must either fit within the size restrictions or support a compliant extension cable which does.

The USB 1.0, 1.1 and 2.0 specifications define two types of connectors for the attachment of devices to the bus: A, and B. The USB 2.0 specification also introduces the mini-B connector, for smaller devices such as PDAs, mobile phones or digital cameras. All connectors are mechanically incompatible, with an A connector always used on the upstream (host) end, and a B connector always used on the downstream (device) end. Hosts and devices include connectors (female) while cables contain plugs (male). Thus all compliant USB cables have an A plug on one end, and either a B or Mini-B on the other end. The A-plug is approximately 4x12 mm, the B-plug is approximately 7x8 mm, and the B-mini plug is approximately 3x7 mm.

However, the mechanical layer has changed in some examples. For example, the IBM UltraPort is a proprietary USB connector located on the top of IBM's laptop LCDs. It uses a different mechanical connector while preserving the USB signaling and protocol. Other manufacturers of small items also developed their own small form factor connector, and a wide variety of these have appeared. For specification purposes, these devices were treated as having a captive cable.

An extension to USB called USB On-The-Go allows a single port to act as either a host or a device - chosen by which end of the cable plugs into the socket on the unit. Even after the cable is hooked up and the units are talking, the two units may "swap" ends under program control. This facility targets units such as PDAs where the USB link might connect to a PC's host port as a device in one instance, yet connect as a host itself to a keyboard and mouse device in another instance. USB On-The-Go has therefore defined two small form factor connectors, the mini-A and mini-B, and a hermaphroditic socket (mini-AB), which should stop the proliferation of proprietary designs.

Wireless USB is a standard being developed to extend the USB standard while maintaining backwards compatibility with USB 1.1 and USB 2.0 on the protocol level.

The maximum length of a USB cable is 5 meters; greater lengths require hubs [1]
(<http://www.usb.org/developers/usbfaq/#cab1>).

Power supply

The USB connector provides a single nominally 5 volt wire from which connected USB devices may power themselves. In practice, delivered voltage can drop well below 5 V, to only slightly above 4 V. The compliance spec requires no more than 5.25 V anywhere and no less than 4.375 V at the worst case; a low-power function after a bus-powered hub. In typical situations the voltage is close to 5 V.

A given segment of the bus is specified to deliver up to 500 mA. This is often enough to power several devices, although this budget must be shared among all devices downstream of an unpowered hub. A bus-powered device may use as much of that power as allowed by the port it is plugged into.

Bus-powered hubs can continue to distribute the bus provided power to connected devices but the USB specification only allows for a single level of bus-powered devices from a bus-powered hub. This disallows connection of a bus-powered hub to another bus-powered hub. Many hubs include external power supplies which will power devices connected through them without taking power from the bus. Devices that need more than 500 mA must provide their own power.

When USB devices (including hubs) are first connected they are interrogated by the host controller, which enquires of each their maximum power requirements. The host operating system typically keeps track of the power requirements of the USB network and may warn the computer's operator when a given segment requires more power than is available (and will generally shut down devices or hubs in order to keep power consumption within the available resource).

A number of devices use this power supply without participating in a proper USB network. The typical example is a USB-powered reading light, but fans, battery chargers (particularly for mobile telephones) and even miniature vacuum cleaners are available. In most cases, these items contain no electronic circuitry, and thus are not proper USB devices at all. This can cause problems with some computers—the USB specification requires that devices connect in a low-power mode (100 mA maximum) and state how much current they need, before switching, with the host's permission, into high-power mode.

Some devices intended for connection to laptops draw more power than is permitted by the specification for a single USB port; to avoid requiring an external power supply, these devices come with dual cables, and the user is instructed that the device must be plugged-into *two* USB ports. On a laptop with only two ports, this means only one such device can be used at a time, unless a powered hub is added. A number of peripherals for IBM laptops (now made by Lenovo) are designed to use dual USB connections in this manner.

USB-powered devices attempting to draw large currents without requesting the power will not work with certain USB controllers, and will either disrupt other devices on the bus or fail to work themselves (or both). Those problems with the abuse of the USB power supply have inspired a number of April Fool hoaxes, like the introduction of a USB-powered George Foreman *iGrill* [2] (<http://www.thinkgeek.com/stuff/looflirpa/igrill.shtml>) and a desktop USB Fondue Set [3] (<http://www.thinkgeek.com/stuff/41/fundue.shtml>).

USB compared to other standards

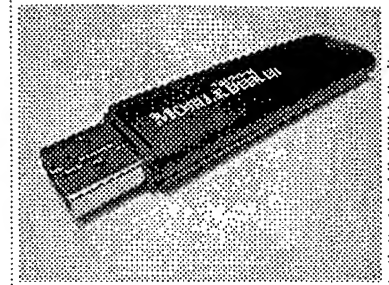
Storage

USB implements connections to storage devices using a set of standards called the *USB mass-storage device class*. This was initially intended for traditional magnetic and optical drives, but has been extended to support a wide variety of devices. USB is not intended to be a primary bus for a computer's internal storage: buses such as ATA (IDE) and SCSI fulfill that role.

However, USB has one important advantage in making it possible to install and remove devices without opening the computer case, making it useful for external drives. Today, a number of manufacturers offer portable USB

hard drives that offer performance comparable to conventional ATA (IDE) drives. These external drives, called enclosures, are often composed of translating devices that connect to USB on one side and to conventional IDE, ATA, ATAPI, or SCSI drives on the other. A drive is installed into the enclosure and the enclosure is then plugged into the computer, thus creating the function of a regular USB mass-storage device.

FireWire technology is also commonly used with portable hard drives, some of which include both USB and FireWire ports. FireWire tends to perform better in speed benchmark tests. The main reason for this is that the tests are conducted point to point (only one device) which means the USB system is always waiting for the drive. Additionally some Operating systems fail to take advantage of the larger transfer sizes in USB2.0 and transfer blocks of 64bytes at a time only (the USB1.1 limit) even though the speed is USB2.0 480 Mbit/s. In a well engineered operating system the USB transfer can sustain rates higher than Firewire but the most common systems are not setup to do this with standard drivers. However, USB ports are more common on consumer-level computers, which enhances the portability of a USB drive. Additionally when multiple devices are connected USB has significant advantages over FireWire,



A Flash Drive, a typical USB mass-storage device

Human-interface devices (HIDs)

USB has not completely replaced AT keyboard connections and PS/2 keyboard and mouse connections, but virtually all PC motherboards manufactured today have one or more USB ports. As of 2004, most new motherboards have multiple USB 2.0 high-speed ports, though some are internal, and require a "header" connection to be accessible from the front or rear of the computer case. Similarly, support for joysticks, keypads, tablets and other human-interface devices is progressively migrating from MIDI, "gameport", and PS/2 connectors to USB. It is now quite common for a mouse or keyboard to be a USB device, which is shipped with a small USB-to-PS/2 adaptor connected to the end of its cable, so it can be used with either USB or PS/2 ports.

Apple computers have used USB mice and keyboards exclusively since January 1999.

FireWire

USB was originally seen as a complement to FireWire, which was designed as a high-speed serial bus which could efficiently interconnect peripherals such as hard disks, audio interfaces, and video equipment. USB originally operated at a far lower data rate and used much simpler hardware, and was suitable for small peripherals such as keyboards and mice.

However, because FireWire ports were more costly to implement than USB ports, primarily due to their per-port licence fee, they were rarely provided as standard equipment on computers, and peripheral manufacturers offered many more USB devices. The introduction of USB 2.0 Hi-Speed, with its widely advertised 480 Mbit/s signalling rate, convinced many consumers that FireWire was outdated (although this was not necessarily the case; see "USB 2.0 Hi-Speed vs FireWire" below).

Today, USB Hi-Speed is rapidly replacing FireWire in consumer products. FireWire retains its popularity in many professional settings, where it is used for audio and video transfer, and data storage.

Technical differences

The most significant technical differences between FireWire and USB include the following:

- USB networks use a tiered-star topology, while FireWire networks use a much less restrictive free-form

topology (which can introduce its own problems). Unlike USB networks, FireWire networks do not require hubs, some devices have two or three ports and act as hub repeaters in the network.

- USB uses a "speak-when-spoken-to" protocol; peripherals cannot communicate with the host unless the host specifically requests communication. A FireWire device can communicate with any other node at any time, subject to network conditions.
- USB supports priority scheduling which can ensure time critical servicing. Firewire devices must compete with all other Firewire devices for every service.
- USB 2.0 supports split transactions which allow the bus to be used by higher speed devices while slower speed devices are still processing requests. In Firewire all devices must wait for slower devices to complete their transaction before continuing.
- A USB network relies on a single host at the top of the tree to control the network. FireWire network relies on a more expensive chip inside each device so that any capable node can control the network.
- A USB network will provide a minimum power to every device. Firewire devices can optionally provide additional power to the network but in general they don't. Bus powered devices are not common in Firewire (like memory sticks).
- All forms of FireWire use significantly different, signalling systems than USB. There are some similarities with the cycle and frame markers respectively. Firewire devices have to indicate both the source and destination; USB only requires the destination because the source is always the host.
- FireWire system performance is affected by the order in which devices are connected. The USB system performance doesn't change with topology -- it is a function of the host and the operating system.

These and other differences reflect the differing design goals of the two busses: USB was designed for simplicity and low cost, while FireWire was designed for high performance, particularly in time-sensitive applications such as audio and video.

USB 2.0 Hi-Speed vs FireWire

The signalling rate of USB 2.0 Hi-Speed mode is 480 megabits per second, while the signalling rate of FireWire 400 (IEEE 1394a) is 393.216 Mbit/s [4] (<http://www.choice.com.au/viewArticle.aspx?id=104527&catId=100274&tid=100008&p=1>). USB can require more host resources than Firewire due to the need for the host to provide the arbitration and scheduling of transactions. USB transfer rates are generally higher than Firewire due to the need for Firewire devices to arbitrate for bus access. A single Firewire device may achieve a transfer rate for Firewire 400 as high as 41MB/s. While for USB 2.0 the rate can be higher 55MB/s (for a single device). In a multi device environment Firewire rapidly loses ground to USB: Firewire's mixed speed networks and long connection chains dramatically affect its performance.

The peer to peer nature of Firewire requires devices to arbitrate, which means a FireWire bus must wait until a given signal has propagated to all devices on the bus. The more devices on the bus the lower the peak performance. Conversely, for USB the maximum timing model is fixed and is limited only by the host-device branch (not the entire network). Furthermore, the host-centric nature of USB allows the host to allocate more bandwidth to high priority devices instead of forcing them to compete for bandwidth as in Firewire. Why then can some USB devices only sustain 34MB/s not 55MB/s? The main reason is usually that the devices themselves are slow and spend most of the time NAK'ing the host to indicate they are not ready - this is particularly true of memory sticks. So the sustained transfer rate is a limitation of the individual device technology not the infrastructure. It is a testament to the flexibility of the USB bus that it can handle wide variances in device

performances. In addition to this some operating systems take a conservative approach to scheduling transactions and limit the number of transfers per frame. Reducing the maximum transfers from say the theoretical 13 per frame to 10 or 9. Therefore if high speed transfer is what you need you should match this with a good host controller and operating system.

In 2003, FireWire was updated with the IEEE 1394b specification. This provides a new mode called S800, which operates at 786.432 Mbit/s. S800 requires a new physical layer, but S800 nodes can be connected to existing FireWire 1394a ports, just as USB Hi-Speed nodes will operate with older full-speed hosts. However unlike USB Hi-Speed systems which can change the speeds on each branch a 1394a device on a 1394b system requires all devices to fall to 1394a speeds. IEEE 1394b also provides rates up to approximately 3.2 Gbit/s; however, the higher rates use special physical layers which are incompatible with 1394a devices.

Version history

USB

- **USB 1.0 FDR:** Released in November 1995, the same year that Apple adopted the IEEE 1394 standard known as FireWire.
- **USB 1.0:** Released in January 1996.
- **USB 1.1:** Released in September 1998.
- **USB 2.0:** Released in April 2000. The major feature of this standard was the addition of high-speed mode. This is the current revision.
- **USB 2.0:** Revised in December 2002. Added three speed distinction to this standard, allowing all devices to be USB 2.0 compliant even if they were previously considered only 1.1 or 1.0 compliant. This makes the backwards compatibility explicit, but it becomes more difficult to determine a device's throughput without seeing the symbol. As an example, a computer's port could be incapable of USB 2.0's hi-speed fast transfer rates, but still claim USB 2.0 compliance (since it supports some of USB 2.0).

USB On-The-Go Supplement

- **USB On-The-Go Supplement 1.0:** Released in December 2001.
- **USB On-The-Go Supplement 1.0a:** Released in June 2003. This is the current revision.

Wireless USB

Released in May 12, 2005 (<http://www.usb.org/developers/wusb/>). Wireless USB uses UWB (<http://wimedia.org/en/index.asp>) (Ultra Wide Band) as the radio technology.

Extensions to USB

The PictBridge standard allows for interconnecting consumer imaging devices. It typically uses USB as the underlying communication layer.

Microsoft's Xbox game console uses standard USB 1.1 signalling in its controllers, but features a proprietary connector rather than the standard USB connector. (However, Microsoft uses standard USB 2.0 connectivity in its newer Xbox 360.) Similarly IBM UltraPort uses standard USB signalling, but uses a proprietary connection format. And [Powered USB] uses standard USB signalling with the addition of extra power lines for Point of sale

terminals.

The USB Implementers Forum is working on a wireless networking standard based on the USB protocol. Wireless USB is intended as a cable-replacement technology, and will use Ultra wideband wireless technology for data rates of up to 480 Mbit/s. Wireless USB is well suited to wireless connection of PC centric devices, just as Bluetooth is now widely used for mobile phone centric personal networks (at much lower data rates). See <http://www.usb.org/developers/wusb/> for more details.

Communication with USB devices

Communication between software and USB devices depends upon the Operating System (Windows, Macintosh, Linux etc) and the language you choose (Java, C++, Delphi etc).

Communication from the Linux OS

- General - <http://www.linux-usb.org/>
- Java - The jUSB (<http://jusb.sourceforge.net/>) project provides a Free Software (and Open Source) Java API for USB, supporting applications using Java host-side software to drive USB devices. This API is unfortunately limited to Linux.

Communication from the Mac OS

- General - apple (<http://developer.apple.com/devicedrivers/usb/index.html>) has this page on General Mac USB Development
- Java - No info is available on this combination.

Communication from the Windows OS

- General - USBIO (<http://www.thesycon.de/eng/usbio.shtml>) has C++ drivers for USB communication on windows from C & C++. Their COM interface allows for Delphi, C# and VB development. Java development is possible via JNI.
- Java - The Mike Stahl (<http://www.steelbrothers.ch/jusb/>) started work on this combination in 2003. The usb.windows package has a partial Windows implementation of a usb.core.Host object, bootstrapping support, and other classes leveraging Windows USB support. It appears that no work has been done on this package since 2003 so it may be abandoned.

Other communication options

If your Operating System and language combination is not supported, another option is a USB to RS-232 bridge. FTDI (<http://www.ftdichip.com/FTDrivers.htm>) Chip provides virtual COM drivers with its chips, to make the USB device look to the host software like a COM (RS-232) port.

See also

- ACCESS.bus
- FireWire (also known as IEEE 1394, or i.link)
- USB Flash Drive
- USB streaming
- U3
- Serial cable (obsoleted by USB and Wi-Fi)

External links

- Home of USB Implementers Forum, Inc. (<http://www.usb.org/>), including the USB 2.0 specification (<http://www.usb.org/developers/docs/>)
- USB Central (<http://www.lvr.com/usb.htm>) for developers of USB devices and hosts
- USB for DOS (<http://www.bootdisk.com/usb.htm>)
- Linux USB Project (<http://www.linux-usb.org/>), containing much technical information and documentation
- USB Networking Introduction (http://www.windowsnetworking.com/articles_tutorials/usbmain.html)
- Linux usbmount (<http://usbmount.alioth.debian.org/>).
- USB in a NutShell (<http://www.beyondlogic.org/usbnutshell/usb-in-a-nutshell.pdf>) - a primer for developers
- Universal Host Controller Interface (UHCI) (<http://developer.intel.com/technology/usb/uhci11d.htm>)

Retrieved from "http://en.wikipedia.org/wiki/Universal_Serial_Bus"

Categories: Computer buses | USB

-
- This page was last modified 08:36, 26 January 2006.
 - All text is available under the terms of the GNU Free Documentation License (see **Copyrights** for details).
Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc.
 - Privacy policy
 - About Wikipedia
 - Disclaimers